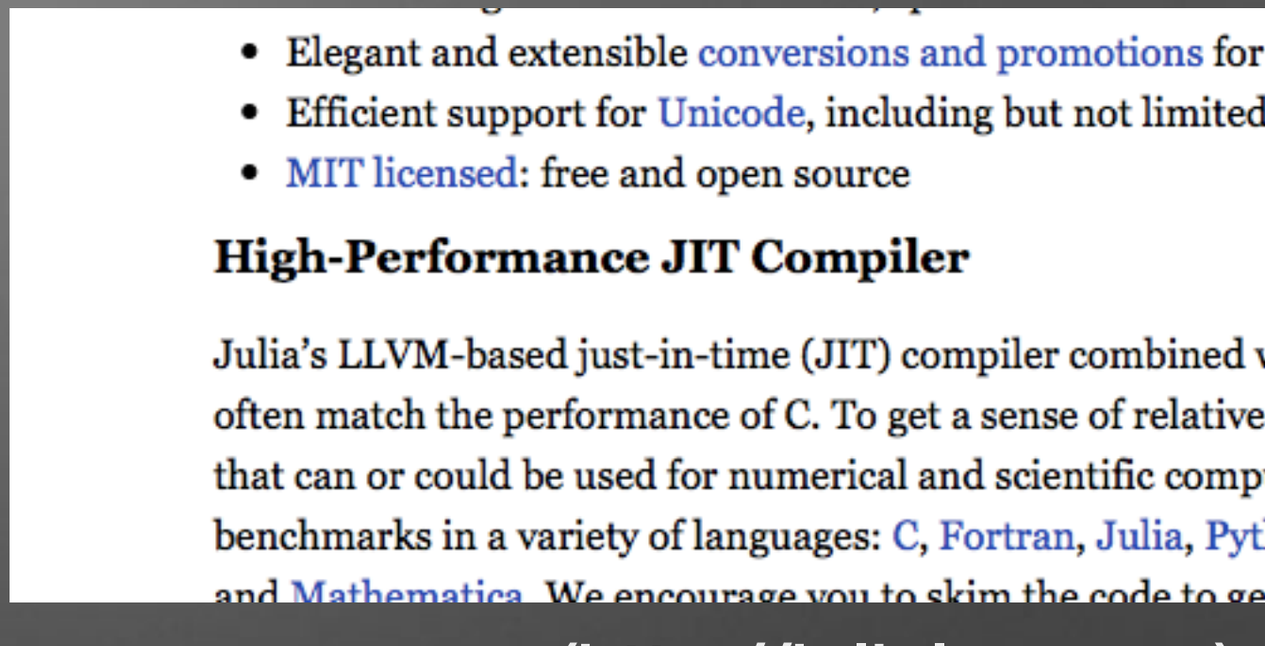
But doesn't it use a JIT?

The homepage of Julia describes the LLVM JIT compiler that helps to make Julia fast.



- Elegant and extensible conversions and promotions for
- Efficient support for Unicode, including but not limited
- MIT licensed: free and open source

High-Performance JIT Compiler

Julia's LLVM-based just-in-time (JIT) compiler combined v
often match the performance of C. To get a sense of relative
that can or could be used for numerical and scientific comp
benchmarks in a variety of languages: C, Fortran, Julia, Pyt
and Mathematica. We encourage you to skim the code to ge

(http://julialang.org)

What it doesn't mention, is that Julia's language design permits better Ahead-of-Time analysis than a traditional JIT must deal with.

What does AoT mean?

Full pre-analysis (type inference) valid

Methods can’t be changed on-stack

No runtime checks on types required by the JIT

Infamous issue #265: previous statements aren’t reliably enforced by the runtime / compiler :(

Adding a method requires updating the AoT compiled code:

- “toplevel” code vs. function code
- natural separation: defining code vs. running data